Comprehensive Bug Bounty Hunting Bootcamp Guide

A complete guide to bug bounty hunting for beginners, covering the top vulnerabilities, tools, techniques, and defensive measures. Designed for Parrot Security OS.

Welcome to

Bug Bounty Hunting

This comprehensive guide will take you from beginner to proficient bug bounty hunter. You'll learn about the most critical vulnerabilities, how to find them, exploit them ethically, and understand how to defend against them.

What You'll Learn

OWASP Top 10

The OWASP Top 10 vulnerabilities in depth

Reconnaissance

Reconnaissance and information gathering techniques

Essential Tools

Essential tools and how to use them on Parrot Security OS

Case Studies

Real-world examples and case studies

Methodology

Ethical hacking methodology

Reporting

How to write effective bug bounty reports

Prerequisites

Basic understanding of web technologies

HTML, HTTP, JavaScript

Familiarity with Linux command line

Comfortable navigating and executing commands

Parrot Security OS installed and configured

Your testing environment ready to go

A curious and ethical mindset

Commitment to responsible disclosure

⚠️ Legal and Ethical Notice

Always obtain explicit permission before testing any system. Only test systems within authorized bug bounty program scopes. Unauthorized access to computer systems is illegal and unethical.

Table of Contents

Part 1: Foundations

1. Bug Bounty Fundamentals
2. Setting Up Your Environment (Parrot Security OS)
3. Reconnaissance Methodology

Part 2: Top Vulnerabilities (OWASP Top 10)

1. Broken Access Control (A01:2021)
2. Cryptographic Failures (A02:2021)
3. Injection Vulnerabilities (A03:2021)
4. Insecure Design (A04:2021)
5. Security Misconfiguration (A05:2021)
6. Vulnerable and Outdated Components (A06:2021)
7. Identification and Authentication Failures (A07:2021)
8. Software and Data Integrity Failures (A08:2021)
9. Security Logging and Monitoring Failures (A09:2021)
10. Server-Side Request Forgery (A10:2021)

Part 3: Additional Critical Vulnerabilities

1. Cross-Site Scripting (XSS)
2. Cross-Site Request Forgery (CSRF)
3. XML External Entity (XXE)
4. Remote Code Execution (RCE)
5. File Upload Vulnerabilities
6. API Security Issues

Part 4: Advanced Topics

1. Bug Bounty Strategy and Methodology
2. Writing Effective Reports
3. Tools Mastery Guide

Part 1

Foundations

Chapter 1: Bug Bounty Fundamentals

What is Bug Bounty Hunting?

Bug bounty hunting is the practice of ethically finding and reporting security vulnerabilities in organizations' systems in exchange for recognition and monetary rewards.

Key Principles

Always work within scope

Only test authorized targets

Be ethical

Never access user data, cause damage, or perform destructive tests

Document everything

Keep detailed notes and screenshots

Communicate clearly

Write professional, detailed reports

Continuous learning

The security landscape is always evolving

Popular Bug Bounty Platforms

HackerOne

Largest bug bounty platform with thousands of programs

Bugcrowd

Wide variety of public and private programs

Intigriti

European-focused with strong community

YesWeHack

Growing platform with diverse programs

Synack

Invite-only, vetted researchers

Bug Bounty Hunter Mindset

Think like an attacker:

- Question every input field and parameter
- Test boundary conditions
- Look for business logic flaws
- Chain multiple low-severity issues into high impact
- Check for information disclosure
- Test authentication and authorization thoroughly

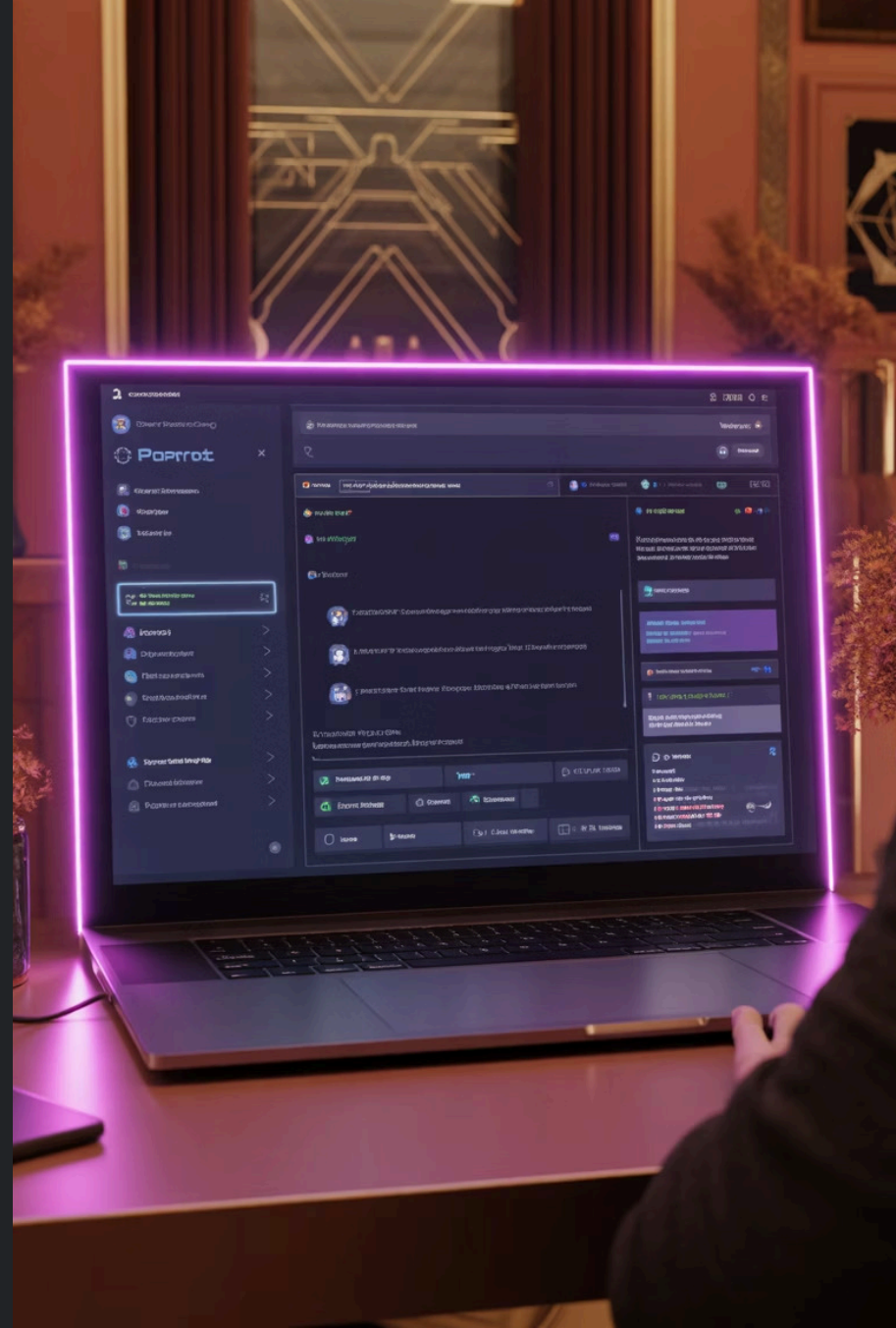
Methodology over tools:

- Understand the "why" behind vulnerabilities
- Develop a systematic approach
- Don't rely solely on automated scanners
- Manual testing finds unique bugs

Chapter 2: Setting Up Your Environment

Why Parrot Security OS?

Parrot Security OS is a Debian-based Linux distribution designed for security testing, digital forensics, and privacy. It comes pre-installed with hundreds of security tools.



Advantages of Parrot Security OS

Lightweight and fast

Optimized performance for security testing

Privacy-focused

Built with security and anonymity in mind

Comprehensive tool suite

Hundreds of pre-installed security tools

Regular updates

Constantly maintained and improved

Active community

Strong support and resources

Essential System Configuration

Update Your System

```
# Update package lists
sudo apt update

# Upgrade all packages
sudo apt upgrade -y

# Update Parrot-specific packages
sudo parrot-upgrade
```

Install Additional Tools

While Parrot comes with most tools pre-installed, you may want to add these:

```
# Caido (Modern web security testing tool)
wget https://caido.io/releases/caido-desktop-latest.deb
sudo dpkg -i caido-desktop-latest.deb

# Go (required for many tools)
sudo apt install golang-go -y

# Python tools
sudo apt install python3-pip -y

# Additional reconnaissance tools
go install -v github.com/projectdiscovery/subfinder/v2/cmd/subfinder@latest
go install -v github.com/projectdiscovery/httpx/cmd/httpx@latest
go install -v github.com/projectdiscovery/nuclei/v2/cmd/nuclei@latest

# Add Go bin to PATH
echo 'export PATH=$PATH:~/go/bin' >> ~/.bashrc
source ~/.bashrc
```

Essential Tools Overview

Reconnaissance Tools

- **Subfinder**

  Subdomain enumeration

- **Amass**

  Comprehensive OSINT framework

- **Assetfinder**

  Fast subdomain discovery

- **httpx**

  HTTP toolkit for probing

- **waybackurls**

  Fetch URLs from Wayback Machine

Web Application Testing Tools

Caido

Modern web security toolkit (alternative to Burp Suite)

OWASP ZAP

Free web application scanner

Nikto

Web server scanner

WhatWeb

Web technology identifier

Fuzzing and Discovery Tools

ffuf

Fast web fuzzer

gobuster

Directory/file brute-forcing

wfuzz

Web application fuzzer

Arjun

HTTP parameter discovery

Network Tools

nmap

Network scanner

masscan

Fast port scanner

netcat

Network utility

Exploitation Tools

sqlmap

Automated SQL injection tool

XSStrike

XSS detection suite

Metasploit Framework

Exploitation framework

Browser Setup

Firefox Configuration

Parrot comes with Firefox ESR. Configure it for security testing:

l. Install Essential Extensions:

- FoxyProxy Standard (proxy management)
- Wappalyzer (technology identification)
- Cookie-Editor (cookie manipulation)
- User-Agent Switcher

2. Configure Proxy:

- Open FoxyProxy
- Add new proxy: 127.0.0.1:8080 (for Caido/ZAP)
- Create patterns for different tools

Installing Caido Certificate

```
# Start Caido
caido

# Navigate to http://127.0.0.1:8080 in Firefox
# Download the CA certificate
# Import into Firefox: Settings > Privacy & Security > Certificates > View Certificates > Import
```

Workspace Organization

Create a structured workspace for your bug bounty activities:

```
# Create main bug bounty directory
mkdir -p ~/bugbounty/{recon,scanning,exploitation,reports,tools,wordlists}
cd ~/bugbounty

# Create subdirectories for each target
mkdir -p targets/{target1,target2,target3}

# Download useful wordlists
cd wordlists
wget https://github.com/danielmiessler/SecLists/archive/master.zip
unzip master.zip
mv SecLists-master SecLists
```

Chapter 3: Reconnaissance Methodology

Reconnaissance is the foundation of successful bug bounty hunting. The more information you gather, the more attack surface you'll discover.

Phase 1: Passive Reconnaissance

Passive recon gathers information without directly interacting with the target.

Subdomain Enumeration

Using Subfinder:

```
# Basic subdomain enumeration
subfinder -d target.com -o subdomains.txt

# Using multiple sources
subfinder -d target.com -all -o subdomains_all.txt

# Silent mode (no banner)
subfinder -d target.com -silent -o subdomains.txt
```

Using Amass

```
# Passive enumeration only
amass enum -d target.com -o amass_results.txt

# With brute force
amass enum -d target.com -brute -o amass_brute.txt

# Active enumeration (more aggressive)
amass enum -active -d target.com -o amass_active.txt
```

Using Assetfinder

```
# Simple subdomain discovery
assetfinder --subs-only target.com > assets.txt
```

Combine and deduplicate results:

```
# Merge all subdomain files
cat subdomains.txt amass_results.txt assets.txt | sort -u > all_subdomains.txt

# Count unique subdomains
wc -l all_subdomains.txt
```

Certificate Transparency Logs

Search for subdomains in CT logs:

```
# Using crt.sh
curl -s "https://crt.sh/?q=%25.target.com&output=json" | jq -r '.[].name_value' | sort -u > ct_subdomains.txt
```

Archive-Based Discovery

Find old endpoints and parameters using web archives:

```
# Using waybackurls
waybackurls target.com > wayback_urls.txt

# Using gau (Get All URLs)
gau target.com > gau_urls.txt

# Filter for interesting files
cat wayback_urls.txt | grep -E "\.js$|\.php$|\.asp$|\.json$" > interesting_files.txt
```

Google Dorking

Use advanced search operators to find sensitive information:

Find login pages

```
site:target.com inurl:login
site:target.com inurl:admin
```

Find exposed files

```
site:target.com filetype:pdf
site:target.com filetype:xls
site:target.com filetype:sql
site:target.com filetype:env
```

Find error messages

```
site:target.com intext:"error" | intext:"warning"
```

Find directory listings

```
site:target.com intitle:"index of"
```

Find exposed configuration files

```
site:target.com ext:xml | ext:conf | ext:cnf | ext:config
```

Find API endpoints

```
site:target.com inurl:api
```

Phase 2: Active Reconnaissance

Active recon involves directly interacting with target systems.

Probing Live Hosts

Using httpx:

```
# Probe for live hosts
cat all_subdomains.txt | httpx -o live_hosts.txt

# Get status codes and response sizes
cat all_subdomains.txt | httpx -status-code -content-length -o live_hosts_detailed.txt

# Take screenshots
cat all_subdomains.txt | httpx -screenshot -o screenshots/

# Extract technologies
cat all_subdomains.txt | httpx -tech-detect -o tech_stack.txt
```

Port Scanning

Using Nmap:

```
# Quick scan with version detection
nmap -sV -sC target.com -oN nmap_scan.txt

# Scan all ports
nmap -p- target.com -oN nmap_all_ports.txt

# Scan specific ports
nmap -p 80,443,8080,8443,8000,8888 target.com -oN nmap_web_ports.txt

# Aggressive scan (OS detection, scripts, traceroute)
nmap -A target.com -oN nmap_aggressive.txt

# Fast scan of multiple hosts
nmap -iL live_hosts.txt -F -oN nmap_fast.txt
```

Using Masscan

```
# Super fast port scan (requires root)
sudo masscan -p1-65535 target.com --rate=1000 -oL masscan_results.txt
```

DNS Enumeration

```
# DNS lookup
dig target.com ANY

# Get all DNS records
dig target.com A
dig target.com AAAA
dig target.com MX
dig target.com TXT
dig target.com NS

# Zone transfer attempt (rarely works)
dig axfr @ns1.target.com target.com

# Using host command
host target.com
host -t mx target.com

# Reverse DNS lookup
host

# WHOIS information
whois target.com
```

Web Technology Fingerprinting

Using WhatWeb:

```
# Basic scan
whatweb target.com

# Aggressive scan
whatweb -a 3 target.com

# Scan multiple URLs
whatweb -i live_hosts.txt -a 3 --log-verbose=whatweb_results.txt
```

Using Wappalyzer (browser extension):

- Automatically detects technologies as you browse
- Shows CMS, frameworks, analytics, CDN, etc.

Using Nikto

```
# Basic web server scan
nikto -h http://target.com

# Save results
nikto -h http://target.com -o nikto_results.txt

# Scan with SSL
nikto -h https://target.com -ssl
```

Directory and File Discovery

Using Gobuster:

```
# Directory brute-forcing
gobuster dir -u http://target.com -w /usr/share/wordlists/dirb/common.txt -o gobuster_dirs.txt

# With file extensions
gobuster dir -u http://target.com -w /usr/share/wordlists/dirb/common.txt -x php,html,js,txt -o gobuster_files.txt

# Faster scan with more threads
gobuster dir -u http://target.com -w ~/bugbounty/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt -t 50 -o
gobuster_fast.txt
```

Using ffuf

```
# Directory fuzzing
ffuf -u http://target.com/FUZZ -w /usr/share/wordlists/dirb/common.txt -o ffuf_dirs.json

# With specific extensions
ffuf -u http://target.com/FUZZ -w wordlist.txt -e .php,.html,.js,.txt

# Filter by status code
ffuf -u http://target.com/FUZZ -w wordlist.txt -fc 404

# Filter by response size
ffuf -u http://target.com/FUZZ -w wordlist.txt -fs 1234

# Match specific response size
ffuf -u http://target.com/FUZZ -w wordlist.txt -ms 5678
```

Using dirsearch

```
# Basic directory search
dirsearch -u http://target.com

# With extensions
dirsearch -u http://target.com -e php,html,js,txt

# Recursive search
dirsearch -u http://target.com -r

# Multiple targets
dirsearch -l live_hosts.txt -e php,html
```

JavaScript Analysis

JavaScript files often contain sensitive information and endpoints:

```
# Extract JavaScript files from wayback results
cat wayback_urls.txt | grep "\.js$" | sort -u > js_files.txt

# Download all JavaScript files
while read url; do
 wget -q "$url" -P js_files/
done < js_files.txt

# Search for API endpoints in JS files
grep -r "api" js_files/
grep -r "endpoint" js_files/
grep -r "http" js_files/ | grep -E "https?://"

# Search for sensitive information
grep -r "password" js_files/
grep -r "api_key" js_files/
grep -r "secret" js_files/
grep -r "token" js_files/
```

Parameter Discovery

Using Arjun:

```
# Discover hidden parameters
arjun -u http://target.com/endpoint

# With specific methods
arjun -u http://target.com/endpoint -m GET,POST

# Using a wordlist
arjun -u http://target.com/endpoint -w params.txt
```

Using ParamSpider:

```
# Extract parameters from wayback data
paramspider -d target.com
```

Reconnaissance Automation Script

Create a bash script to automate your recon workflow:

```
#!/bin/bash
# recon.sh - Automated reconnaissance script

if [ $# -eq 0 ]; then
 echo "Usage: ./recon.sh "
 exit 1
fi

DOMAIN=$1
OUTPUT_DIR="recon_$DOMAIN"

mkdir -p $OUTPUT_DIR
cd $OUTPUT_DIR

echo "[+] Starting reconnaissance for $DOMAIN"

# Subdomain enumeration
echo "[+] Enumerating subdomains..."
subfinder -d $DOMAIN -silent -o subdomains_subfinder.txt
amass enum -passive -d $DOMAIN -o subdomains_amass.txt
assetfinder --subs-only $DOMAIN > subdomains_assetfinder.txt
```

Automation Script (continued)

```
# Combine and deduplicate
cat subdomains_*.txt | sort -u > all_subdomains.txt
echo "[+] Found $(wc -l < all_subdomains.txt) unique subdomains"

# Probe for live hosts
echo "[+] Probing for live hosts..."
cat all_subdomains.txt | httpx -silent -o live_hosts.txt
echo "[+] Found $(wc -l < live_hosts.txt) live hosts"

# Port scanning
echo "[+] Scanning ports..."
nmap -iL live_hosts.txt -F -oN nmap_results.txt

# Technology detection
echo "[+] Detecting technologies..."
cat live_hosts.txt | httpx -tech-detect -silent -o technologies.txt

# Directory discovery on main domain
echo "[+] Discovering directories..."
gobuster dir -u http://$DOMAIN -w /usr/share/wordlists/dirb/common.txt -q -o directories.txt

echo "[+] Reconnaissance complete! Results saved in $OUTPUT_DIR"
```

Make Script Executable

```
chmod +x recon.sh
./recon.sh target.com
```

Part 2

Top Vulnerabilities

OWASP Top 10

OWASP Top 10 (2021)

01

Broken Access Control

A01:2021

02

Cryptographic Failures

A02:2021

03

Injection

A03:2021

04

Insecure Design

A04:2021

05

Security Misconfiguration

A05:2021

OWASP Top 10 (continued)

01

Vulnerable and Outdated Components

A06:2021

02

Identification and Authentication Failures

A07:2021

03

Software and Data Integrity Failures

A08:2021

04

Security Logging and Monitoring Failures

A09:2021

05

Server-Side Request Forgery

A10:2021

A01:2021

Broken Access Control

The most critical vulnerability category, where restrictions on what authenticated users can do are not properly enforced.

A02:2021

Cryptographic Failures

Failures related to cryptography which often lead to exposure of sensitive data.

A03:2021

Injection Vulnerabilities

User-supplied data is not validated, filtered, or sanitized by the application, allowing attackers to inject malicious code.

A04:2021

Insecure Design

Missing or ineffective control design, representing different weaknesses expressed as "missing or ineffective control design."

A05:2021

Security Misconfiguration

Missing appropriate security hardening across any part of the application stack, or improperly configured permissions.

A06:2021

Vulnerable and Outdated Components

Using components with known vulnerabilities, unsupported or out of date software.

A07:2021

Identification and Authentication Failures

Confirmation of the user's identity, authentication, and session management is critical to protect against authentication-related attacks.

A08:2021

Software and Data Integrity Failures

Code and infrastructure that does not protect against integrity violations, such as insecure deserialization.

A09:2021

Security Logging and Monitoring Failures

Without logging and monitoring, breaches cannot be detected, escalated, or responded to effectively.

A10:2021

Server-Side Request Forgery (SSRF)

SSRF flaws occur when a web application fetches a remote resource without validating the user-supplied URL.

Additional Critical Vulnerabilities

Cross-Site Scripting (XSS)

Cross-Site Request Forgery (CSRF)

XML External Entity (XXE)

Remote Code Execution (RCE)

File Upload Vulnerabilities

API Security Issues

Coming Up Next

Deep Dive into Each Vulnerability

In the following sections, we'll explore each vulnerability in detail, including:

- How to identify them
- Exploitation techniques
- Real-world examples
- Defensive measures
- Bug bounty tips

Ready to Start Your Bug Bounty Journey?

Continue to the next sections to master each vulnerability type and become a successful bug bounty hunter.